

Kurzübersicht zur Vorlesung Software Engineering WI 2022 – Kurs B

Prof. Dr.-Ing. habil. Andreas Schmietendorf

Professur Wirtschaftsinformatik/Systementwicklung – HWR Berlin
Privatdozentur Software Engineering – OvG Universität Magdeburg



Abb. erstellt mit Hilfe von KI: <https://playground.com/create>
Prompt: Software Engineering, Computer Programs, IT Architecture, Quality Assurance,
Project Management (generiert am 26.02.2024)

Februar/März 2024

Inhalt

Inhaltliche Ausrichtung der Vorlesung	3
Inhalte der Vorlesung	5
Bewertung der Semesterleistung.....	7
Terminübersicht	8
Hinweise zur Protokollierung der 4 Übungen	9
Quellenverzeichnis	10

Inhaltliche Ausrichtung der Vorlesung

Mit Hilfe des Software-Engineerings sollen Software-Produkte, entsprechend den Prinzipien klassischer Ingenieurdisziplinen (z.B. Bauwesen, Elektrotechnik, Maschinenbau) im Sinne einer industriellen Fertigung erstellt werden. Eine relative frühe Definition zum Software Engineering findet sich unter [IEEE 1990]:

„Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, maintenance, and retirement of software, that is, the application of engineering to software“¹

Eine detailliertere und aktualisierte Auseinandersetzung findet sich unter [Lackes/Siepermann 2018]. Dem entsprechend handelt es sich beim Software Engineering um eine Disziplin die sich mit der Entwicklung, dem Einsatz und der Wartung von Software beschäftigt. In Anlehnung an [Lackes/Siepermann 2018] lassen sich die folgenden Sachverhalte unterscheiden:

1. Ziele: Beherrschung der Problemkomplexität durch die Bereitstellung von Prinzipien, Methoden und Werkzeuge.
2. Einordnung: Diesbezüglich existierenden mehrere Ansätze, die das Software Engineering u.a. dem Gebiet der Wirtschaftsinformatik aber auch der praktischen Informatik zuordnen.
3. Teilgebiete (Fokus: Prozess, Produkt und Ressource):
 - a. Softwareentwicklung in Bezug auf den Lebenszyklus eines Softwaresystems (typ. Problemdefinition, Analyse, Spezifikation, Entwurf, Implementierung, Test, Einsatz, Wartung, Ablösung).
 - b. Projektmanagement in Bezug auf die Schätzung bzw. Ermittlung von Projektumfängen (-kosten), die Projektorganisation, das Versions- und Buildmanagement aber auch das Risikomanagement.
 - c. Softwarequalitätssicherung bezüglich festzulegender Qualitätsmerkmale („nicht funktionale Anforderungen“) bzw. der konstruktiven und analytischen Qualitätssicherung (allg. Softwaretests).
 - d. Softwaretechnologie in Bezug auf die Bereitstellung von Prinzipien, Methoden und Werkzeuge für die Systementwicklung bzw. Programmierung.

¹ IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.121990 (Revision and redesignation of IEEEstd 792-1983)

4. Praxis: Im Mittelpunkt des praktischen Software Engineering stehen Aspekte des Projektmanagements bzw. die Verwendung vielfältig zur Verfügung stehender Werkzeuge.

Grundsätzlich bieten die vorgenannten Ansätze eine gute Orientierung für eine detaillierte und vor allem wissenschaftliche Auseinandersetzung mit dem doch recht breit gefächerten Themengebiet des Software-Engineerings.

Unter Berücksichtigung der durch [Sommerville 2018] behandelten Themengebiete zum Software Engineering bleiben allerdings vielfältige Sachverhalte bei der vorgestellten Definition außen vor. Aus Sicht des Autors werden z.B. die Themengebiete der Softwarearchitektur und der Softwaremodellierung (Komplexitätsbeherrschung durch Abstraktion) maximal impliziert berührt. Ebenso unberücksichtigt bleibt der Aspekt einer zunehmenden Automation der Softwareentwicklung selbst bzw. die agile Ausrichtung der Integration von Softwareentwicklungsprozessen mit betrieblich orientierten Sachverhalten (vgl. Begriff „DevOps“ - Development and Operations).

Die Inhalte der Vorlesung orientieren sich an der Verwendung von Theorien, Methoden, Modellen und Werkzeugen, die eine zunehmende Automatisierung der durchzuführenden Aufgaben im Software Engineering unterstützen. Gewährleistet werden diese Möglichkeiten durch die Berücksichtigung messtechnisch gewonnener Erkenntnisse sowie die Orientierung an Erfahrungen (z.B. Architektur-Pattern) und Standards (z.B. ISO 25000) bzw. letztendlich die Bereitstellung vielfältiger Werkzeuge welche die eigentliche Automatisierung über den gesamten Software-Lebenszyklus unterstützen. Auf dieser Grundlage soll die zeitlich und aufwandsseitig determinierte Bereitstellung von Software, entsprechend den qualitativen und funktionalen Anforderungen der jeweiligen Kundenseite, ermöglicht werden. Im Rahmen dieser Lehrveranstaltung werden dafür die folgenden Themenkomplexe aufgegriffen:

- Software-Management
- Software-Qualitätssicherung
- Software-Architekturen

Bitte berücksichtigen das die aktuell eingestellten Inhalte im laufenden Semester noch angepasst werden können!

Inhalte der Vorlesung

Themenkomplex des Software-Managements:

1. Produktivität in der Softwareentwicklung
 - Begriffsbestimmung
 - Einflussfaktoren auf die Produktivität
 - Möglichkeiten zur Bestimmung des funktionalen Umfangs
2. Konfigurations-, Build- und Wissensmanagement
 - Konfigurationselemente und Versionskontrolle
 - Systemerstellung (Build-Management)
 - Cloud-basierte Entwicklungsumgebungen
3. Prozess- und Vorgehensmodelle
 - Aufgaben und Ziele eines Vorgehensmodells
 - Klassische und agile Ansätze
 - Agile Werte, Prinzipien und Techniken (Manifest)

Optionale Themen:

- Risikomanagement entsprechend PMBOK
- Offshore Projektorganisationen im Software Engineering

Themenkomplex der Software-Qualitätssicherung:

4. Statische Testverfahren zur Bewertung der Produktqualität
 - Überblick zum IEEE Standard 1028 für Reviews
 - Möglichkeiten Inspektion, Review und Walkthrough
 - Ansätze für eine Werkzeugunterstützung
5. Dynamische Testverfahren zur Bewertung der Produktqualität
 - Strukturtests (White Box)
 - Funktionale Tests (Black Box)
 - Agile Testansätze und Test-Standard ISO 29119
6. Generative AI im Software-Engineering
 - Unterstützte Aufgaben (Kommentierung, Analyse, Codefragmente, ...)
 - Umsetzung mit LLMs und Co (z.B. GitHub Copilot)
 - Problembereiche und Risiken

Optionale Themen:

- Verwendung von Softwaremetriken
- Bewertung der Prozessqualität (u.a. ISO 9000, TQM und CMMI)

Themenkomplex der Software-Architekturen:

7. Grundlagen von Softwarearchitekturen
 - Architekturmerkmale und -sichten
 - Qualitätsattribute einer Architektur
 - Entwurfsprinzipien einer Architektur (Pattern, ...)
8. Cloud-basierte Software-Architekturen
 - Grundlagen zum Cloud Computing (IaaS, PaaS, SaaS)
 - Einschätzungen zum Cloud-Computing
 - Cloudbasierte Lösungen
9. Low Code basierte Ansätze
 - Treiber, Vor- und Nachteile, Risiken
 - Prozess und Kriterien zur Plattformauswahl
 - Einsatzszenarien konkreter Werkzeuge (z.B. KNIME)

Optionale Themen:

- Nachhaltige Architekturen mit „Domain Driven Design“

Die als optional ausgewiesenen Themenstellungen werden durch entsprechende Scripte begleitet, so dass bei Interesse ein entsprechende Selbststudium ermöglicht wird. Sofern es der zeitliche Rahmen zulässt, kann auf ausgewählte Sachverhalte bzw. ggf. auftretende Fragen eingegangen werden.

Bewertung der Semesterleistung

Die Bewertung des Faches Software Engineering erfolgt mit Hilfe von 4 über Moodle einzureichenden Übungsaufgaben (d.h. Protokolle), die teamorientiert (maximal 4 Mitglieder) zu bearbeiten sind. Die 4 Übungsaufgaben müssen mindestens jeden der Themenkomplexe unserer Vorlesung einmal erfassen! Dabei handelt es sich um die Themenkomplexe Software-Management, Software-Qualitätssicherung und Software-Architektur. Zu mindestens 2 Übungsaufgaben ist eine Präsentation von ca. 10 min. zzgl. Fragen zu halten, dem entsprechend ergibt sich die Bewertung aus 6 gleichgewichteten Teilleistungen. Im Rahmen der Präsentation können alle Teammitglieder zu den bearbeiteten Inhalten befragt werden, dem entsprechend ist die Anwesenheit zwingend! Bei ggf. sichtbaren Leistungslücken erfolgt eine differenzierte Bewertung der Teammitglieder.

Die Bewertung der Präsentation erfolgt über die Kriterien fachlicher Inhalt, Aufgabentreue, Komplexität, kritische Reflektion, Stringenz und Struktur der Darstellung, Form und Präsentationsstil, Zeittreue und Teamorientierung. Darüber hinaus geht der Umgang mit den gestellten Fragen in die Bewertung ein.

Terminübersicht

Die folgende Terminübersicht gibt eine grobe Orientierung zu den je Vorlesung behandelten Themenstellungen. Aufgrund von ggf. durchgeführten Schwerpunktsetzungen bzw. den durch die Studierenden zu erbringenden Übungsaufgaben kann es hier zu Verschiebungen kommen.

27.02.2024 – Einführung und Produktivität in der SW-Entwicklung

05.03.2024 – Konfigurations- und Buildmanagement

12.03.2024 – Prozessmodelle der SW-Entwicklung – Scrum, XP, ...

19.03.2024 – Statische Softwaretestverfahren – Reviews, ...

26.03.2024 – Dynamische Softwaretestverfahren – Unit-Tests, ...

02.04.2024 – Einsatz generativer KI im Software Engineering

12.04.2024 – Grundlagen/Bedeutung von Softwarearchitekturen

19.04.2024 – Cloud-basierte Softwarearchitekturen

23.04.2024 – Möglichkeiten einer Low-Code basierten Entwicklung

29.04.2024 – Abschlusspräsentationen und Reflektion

Hinweise zur Protokollierung der 4 Übungen

Grundsätzlich bedarf die wissenschaftliche Bearbeitung eines Themengebiets der exakten Protokollierung. Entsprechende Methoden des wissenschaftlichen Arbeitens beziehen sich z.B. auf Umfragen und Experteninterviews, Kriterienbasierte Analysen, die modelltechnische und simulative Erfassung realer Sachverhalte, kontrollierte Experimente, durchgeführte Gruppendiskussionen oder auch Literaturrecherchen. Die Protokollierung dient der Nachvollziehbarkeit und ggf. der benötigten Reproduzierbarkeit erzielter Ergebnisse. Im Allgemeinen werde dafür die folgenden Angaben benötigt:

- Allgemeine Informationen
 - Versuch, Beteiligte Studenten, Datum
 - Rahmenbedingungen (eingesetzte Software, verwendete Hardware, genutztes Netzwerk, Lokationen, ...)
 - Methodisches Vorgehen zu Bearbeitung (Aufgabenverteilung, Hinweise zu verbrauchten Ressourcen, ...)
- Aufgaben des Laborversuchs
 - Ergebnisse entsprechend den Aufgabenstellungen der Übungsaufgaben, ggf. definierte Abgrenzungen und Annahmen.
 - Textliche oder auch tabellarische Ausführungen zu den Lösungen, so dass diese durch einschlägig vorgebildete Wissenschaftlicher und Ingenieure nachvollzogen werden können.
 - Verwendung von Grafiken, Diagrammen und Screenshots zur Verdeutlichung von gewonnenen Erkenntnissen oder auch ggf. auch identifizierten Problemstellungen.
- Zusammenfassung (kritische Bewertung der erreichten Ergebnisse und identifizierter Forschungsbedarf)
- Genutzte Quellen (z.B. Literatur, Internet, ...)

Quellenverzeichnis

[Lackes/Siepermann 2018] Lackes, R.; Siepermann, M.: Software Engineering - Ausführliche Definition im Online-Lexikon, <https://wirtschaftslexikon.gabler.de/definition/software-engineering-42515/version-265861>, letzter Zugriff: 26. Februar 2024

[Sommerville 2018] Sommerville, I.: Software Engineering, 10. Aktualisierte Auflage Pearson Studium

[IEEE 1990] 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology